

# SA3 Final Project

Milestone 2

# HandShake

**Fixing bugs**  
**Polishing code**  
**Finishing features**



# ⊕ Handshake Planning

Milestone 2

Milestone 3

Milestone 4

Milestone 5

+ New view

Filter by keyword or by field

## Todo 3

Draft

Organize the frontend router structure

Draft

Make login and sign up actually work

Draft

Add about us page

## In Progress 2

Draft

Organize the frontend socket structure

Draft

Structure and implementation of socket server

## Done 7

handshake #33

Setup of the Vue frontend

Draft

Organize the message protocol

Draft

Define the structure of the messages

Draft

Complete the documentation for sockets

Draft

Complete the documentation for routes

handshake #30

[BUG] The frontend is not responsive (M2/M3)

Draft

Complete routes

+ Add item

+ Add item

+ Add item

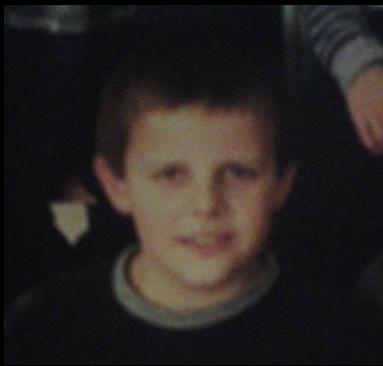
# Roles



CSS + About



Frontend + Backend



Setup + Frontend +  
Tester



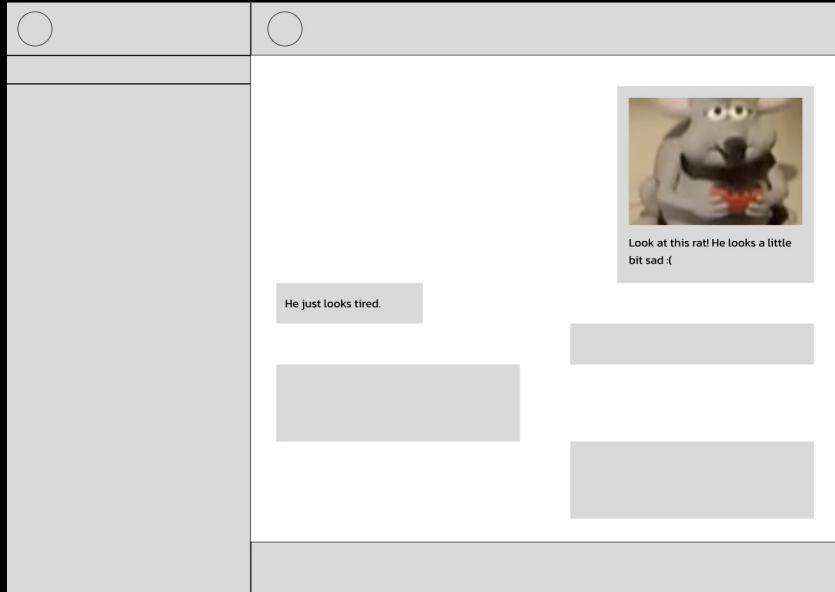
Frontend



Backend + Customer  
+ Tester

# layout





**HandShake**

About us

**Already a user?**

Username

Password

**Log in**

**Join HandShake now.**

Username  E-mail

Password  Confirm password

**Sign up**

made with  by the ogs @ USI



Already a user?

Join HandShake  
now.

Username

Username

Password

Password

Log in

Confirm password

Sign up

Already a user?

Username

Password

Log in

Join HandShake now.

Username

E-mail

Password

Confirm password

Sign up

# [BUG] The frontend is not responsive (M2/M3) #30

Closed

micheledallerive opened this issue 6 days ago · 1 comment



micheledallerive commented 6 days ago • edited

Member

...

## Describe the bug

- The frontend components are not responsive (width/heights set with `vw`).
- The spaces are created using `<br/>` elements (instead of css `gap`)
- The two main sections (chats list and chat) could be implemented as `col-*` (for example, the left column could be `col-12 col-md-4 col-lg-3`, to make it 100% for smartphones, 33% for tablets and 25% for computer and bigger devices. It could be useful also a `max-width` property)
- The Bootstrap classes could be used to make the code easier to read, adapt and understand, since the majority of the components are unique and creating a specific class is not useful.

## To Reproduce

Steps to reproduce the behavior:

1. Take the code of the views.
2. Put them together in a temporary HTML file.
3. Open the file.

## Target

Milestone 2/3(?)

## Screenshots

About us

# HandShake

Already a user?

 Username Password[Log in](#)

Already a user?

 Username Password[Log in](#)

Join HandShake now.

 Username E-mail Password Confirm password[Sign up](#)

Join HandShake now.

 Username E-mail Password Confirm password[Sign up](#)

The background of the image is a dark, moody photograph of oranges. In the foreground, there's a single orange that has been peeled, showing its segments. Behind it are several whole oranges, some with their green leaves still attached. The lighting is low, creating deep shadows and highlighting the texture of the fruit.

# About us / Homepage





LottieFiles



# Lottie Player

# Interactivity Guide

This is a quick demo for using the Lottie web player to add interactivity to your applications





A close-up photograph of a person's hand reaching towards a door handle. The hand is palm-up, with the fingers slightly spread. A keychain hangs from the thumb, featuring a silver metal key and a gold-colored house-shaped key fob. The background is a plain, light-colored wall.

**frontend**

# Frontend #Vue

## Created Login and SignUp

**Already a user?**

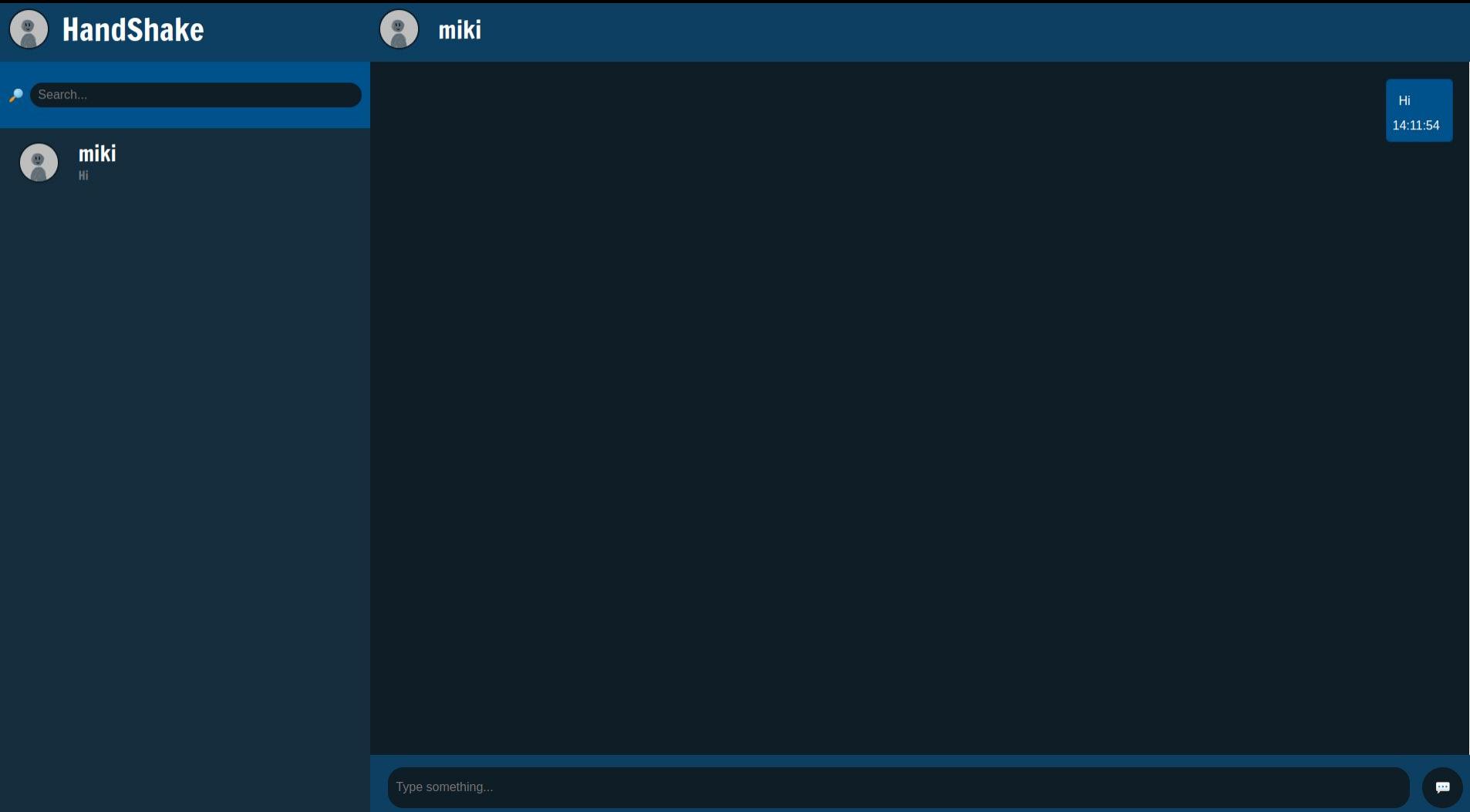
[Don't have an account?](#) [Sign up](#)

**Join HandShake now.**

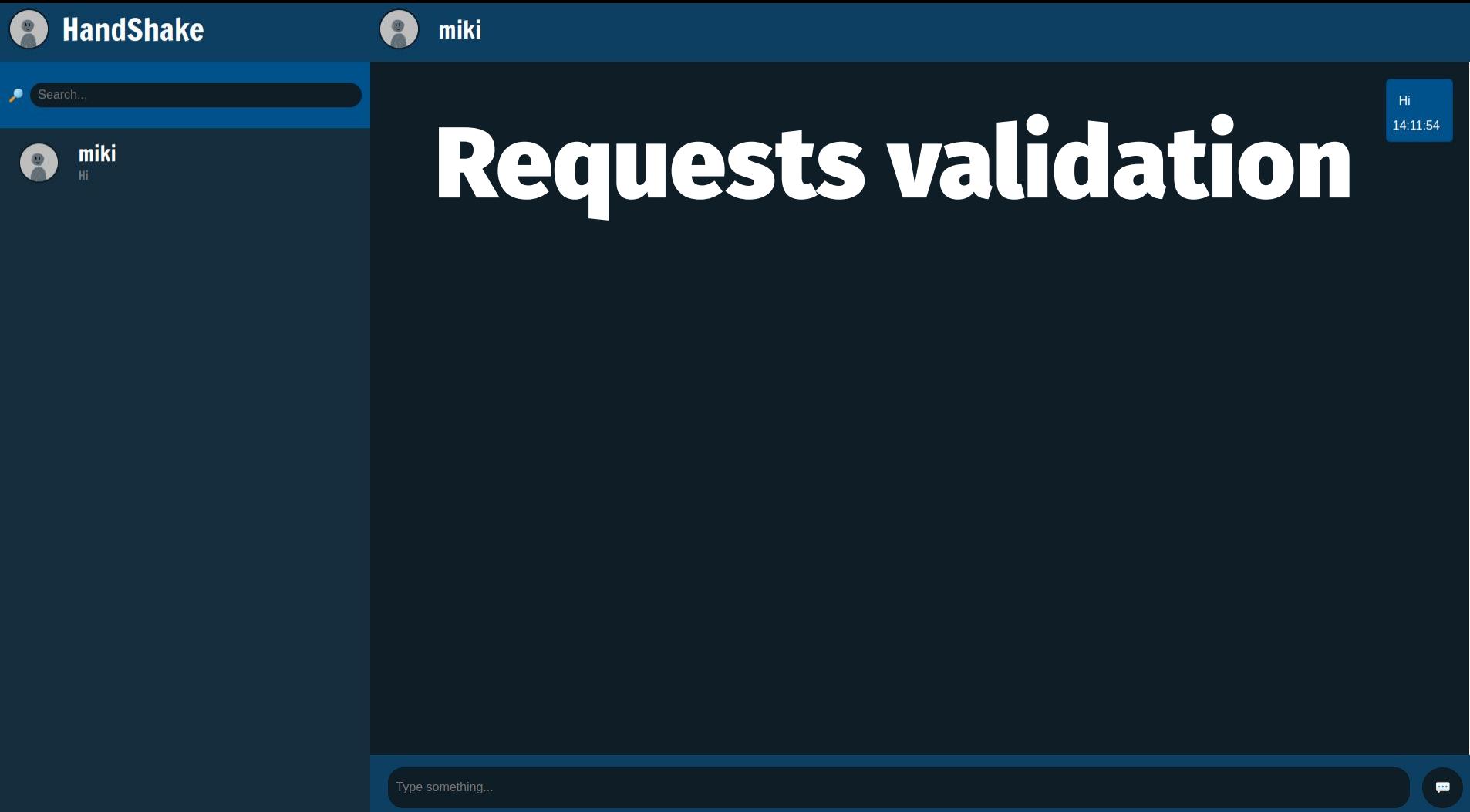
   
   

[Already a user?](#) [Login](#)

# Handled the exchange of messages



# Handled the exchange of messages



# Handled the exchange of messages

The image shows a dark-themed messaging application interface. At the top, there are two user profiles: 'HandShake' on the left and 'miki' on the right. A search bar is located above the profiles. On the far right, a blue message bubble contains the text 'Hi' and the timestamp '14:11:54'. The main area of the screen features a large, semi-transparent gray overlay with the text 'Requests validation' in a large, bold, sans-serif font. Below this overlay, the text 'Check if chat exists' is displayed in a large, bold, white font. At the bottom of the screen, there is a dark blue footer bar with a white input field containing the placeholder 'Type something...' and a small speech bubble icon on the right side.

HandShake

miki

Search...

Hi  
14:11:54

Requests validation

Check if chat exists

Type something...

# Handled the exchange of messages

The image shows a dark-themed messaging application interface. At the top, there's a header bar with the title "Handled the exchange of messages". Below the header, the main content area features large, bold text: "Requests validation", "Check if chat exists", and "Create messages". On the left side, there's a sidebar with a profile picture and the name "HandShake". The main conversation area shows a message from "miki" with the text "Hi" and the timestamp "14:11:54". At the bottom, there's a message input field with the placeholder "Type something..." and a send button icon.

HandShake

miki

Search...

miki

Hi  
14:11:54

## Requests validation

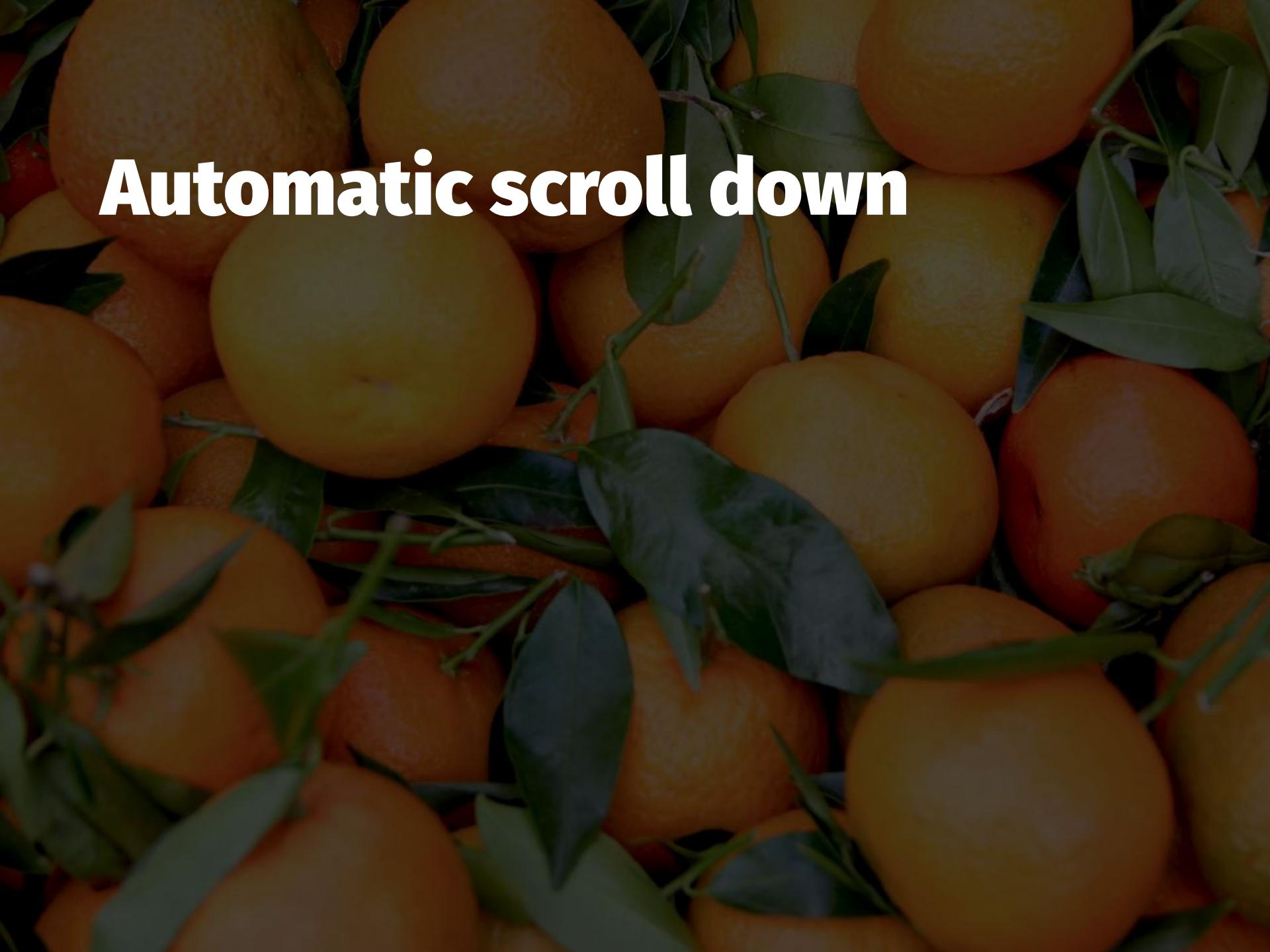
## Check if chat exists

## Create messages

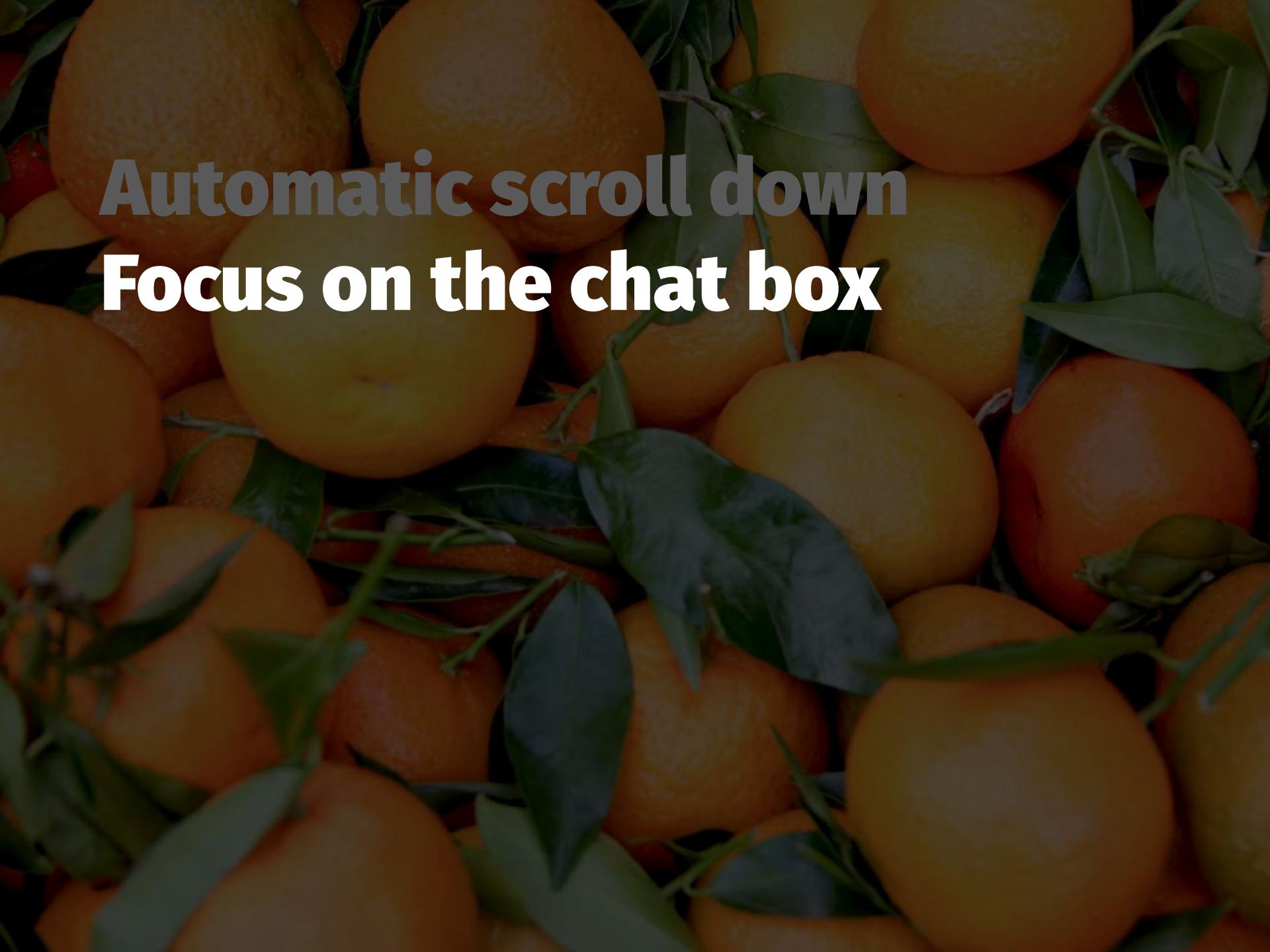
Type something...



# chat features

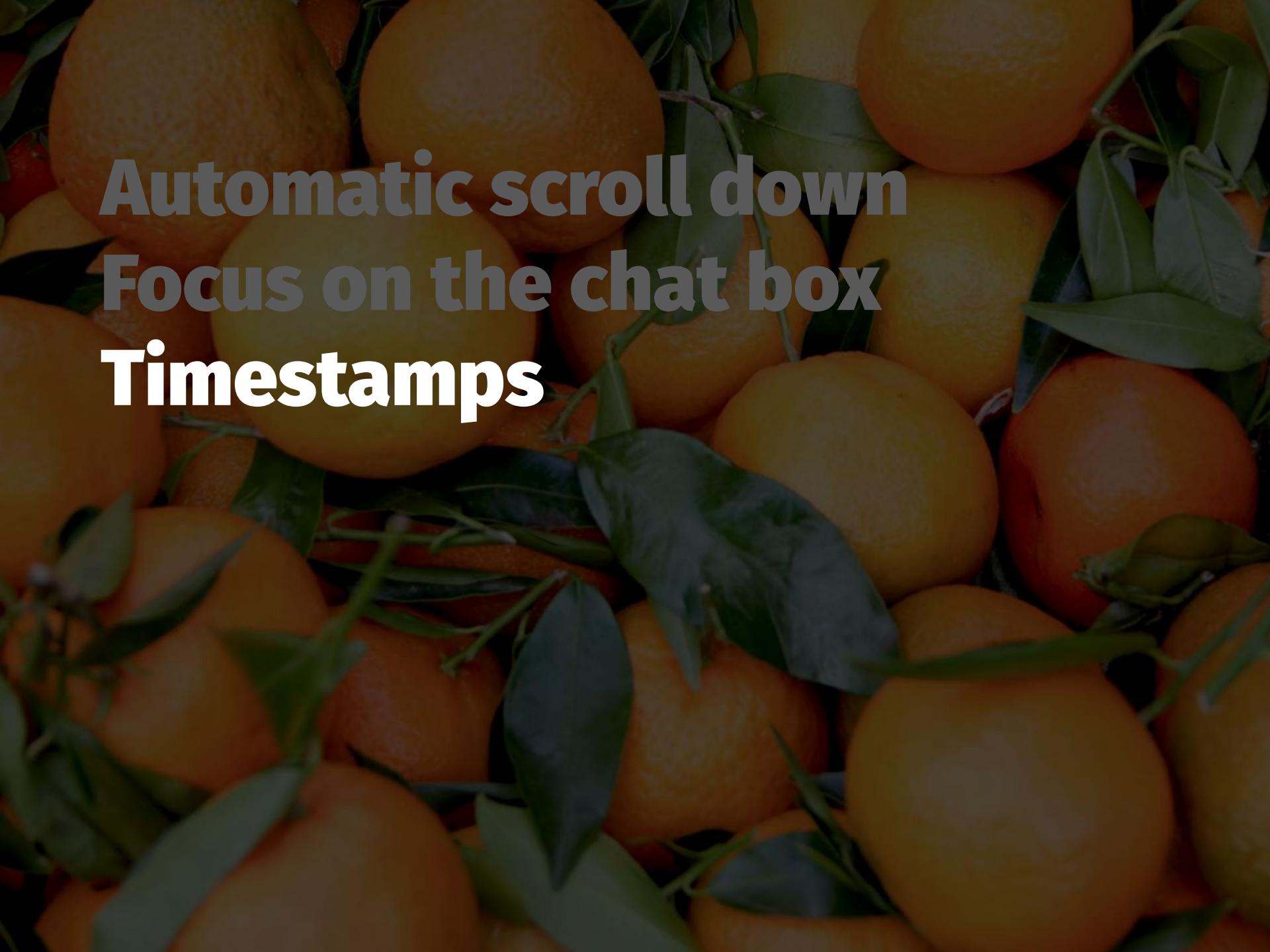
A close-up photograph of a large pile of ripe oranges. The oranges are bright orange with some green stems and leaves visible. They are piled high, filling the frame.

# Automatic scroll down

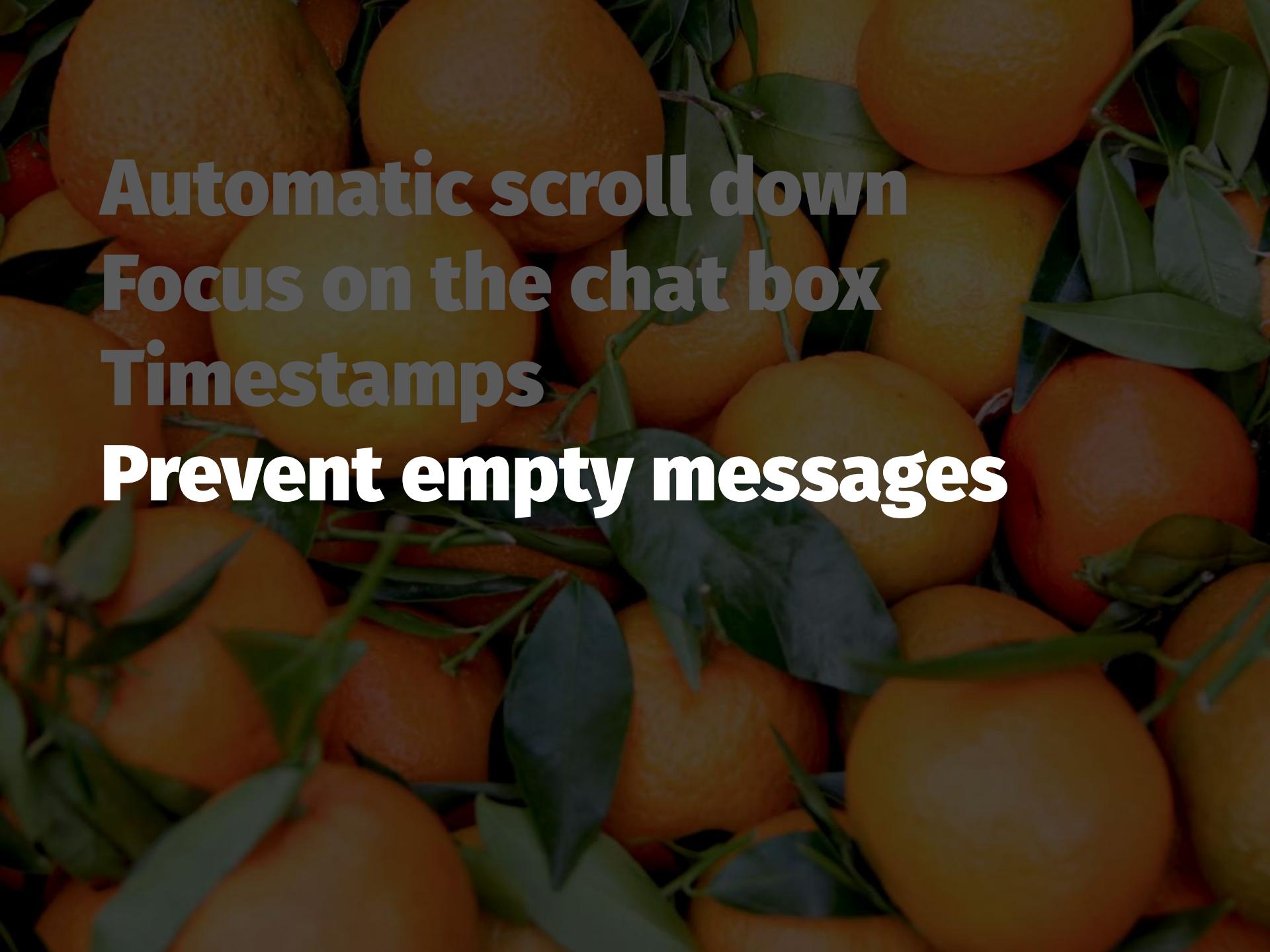


**Automatic scroll down**

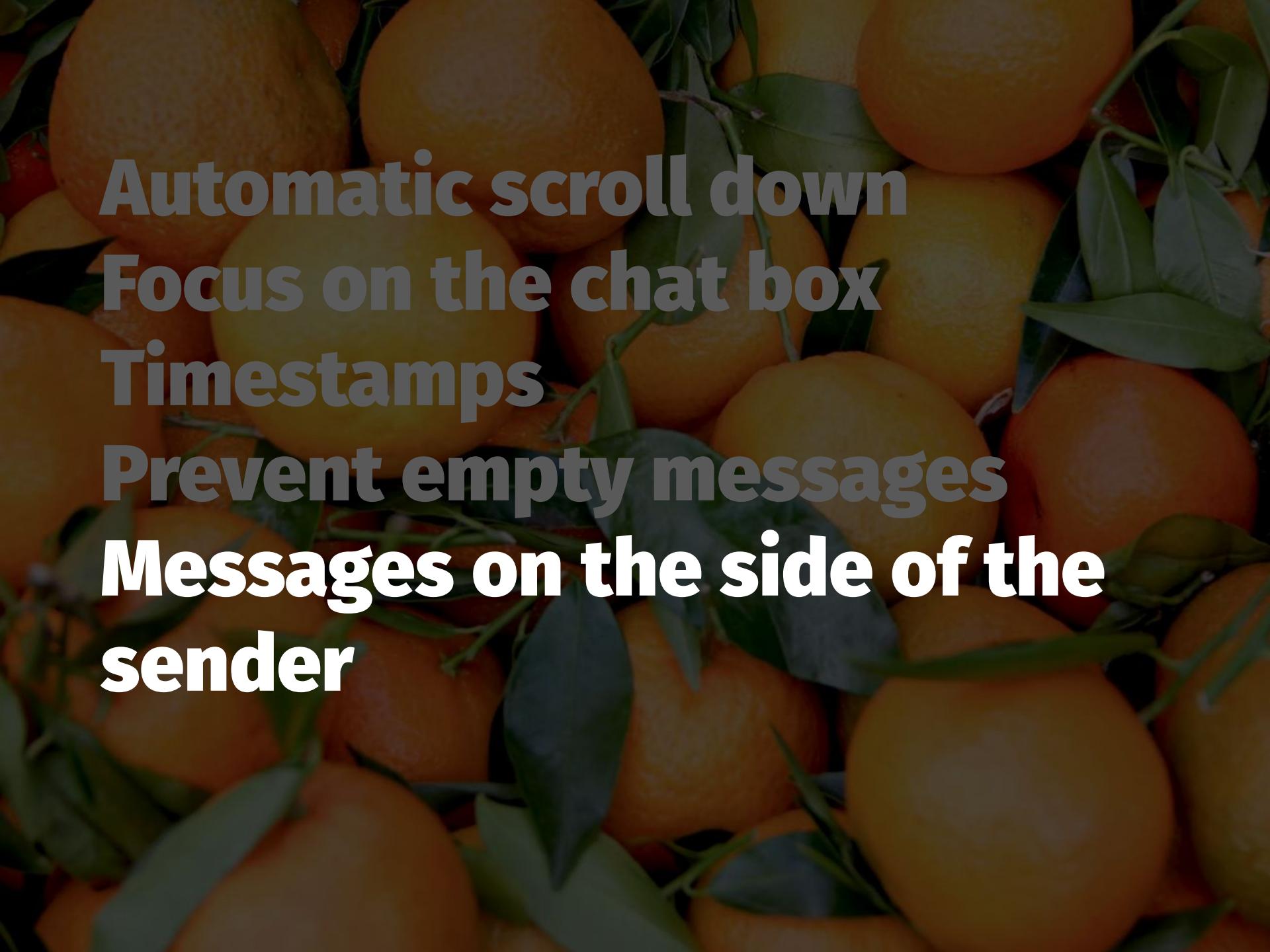
**Focus on the chat box**



**Automatic scroll down  
Focus on the chat box  
Timestamps**



**Automatic scroll down**  
**Focus on the chat box**  
**Timestamps**  
**Prevent empty messages**



**Automatic scroll down**  
**Focus on the chat box**  
**Timestamps**  
**Prevent empty messages**  
**Messages on the side of the  
sender**

**Automatic scroll down**  
**Focus on the chat box**  
**Timestamps**  
**Prevent empty messages**  
**Messages on the side of the**  
**sender**  
**Fixed bugs**

# Message protocol

# Message exchange dilemma: HTTP / socket



# **Message exchange dilemma: HTTP / socket**

**Socket as passive state  
updates propagation for clients**

# **Message exchange dilemma: HTTP / socket**

**Socket as passive state  
updates propagation for clients**

**Diving HTTP routes and socket**

# **Message exchange dilemma: HTTP / socket**

**Socket as passive state  
updates propagation for clients**

**Diving HTTP routes and socket**

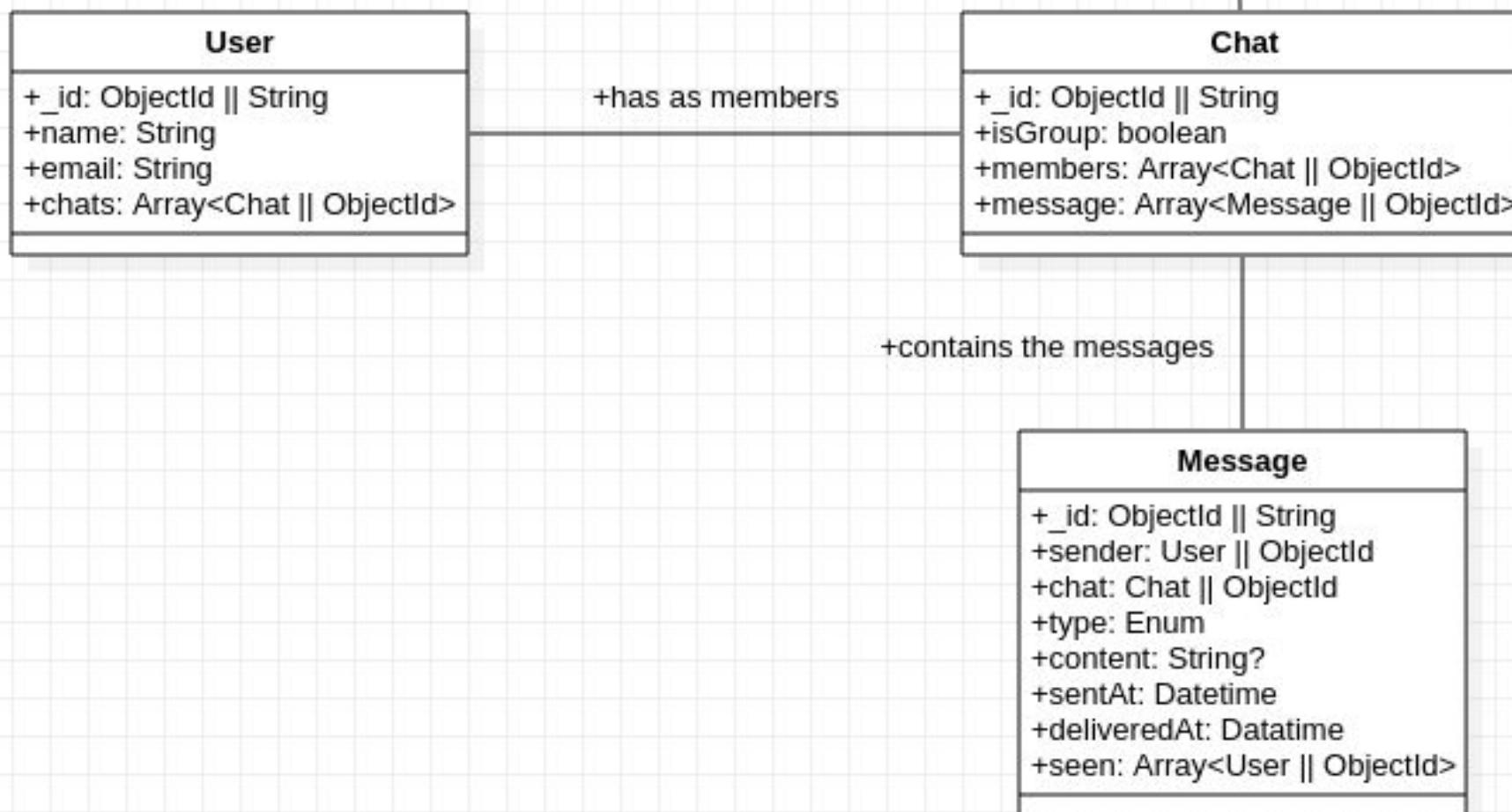
**Prototyping**

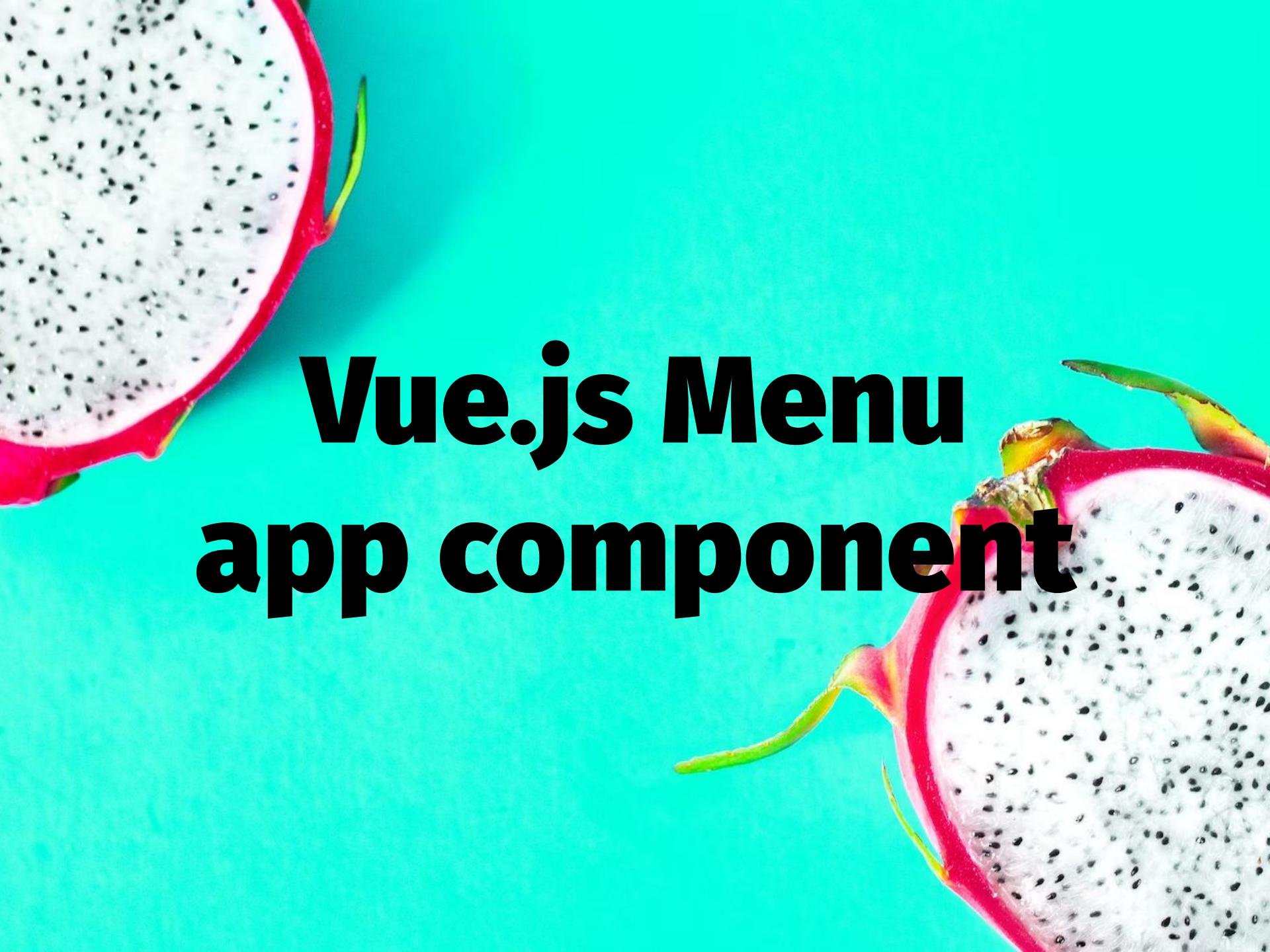


# UML

Message.content is always a string.

It contains either the string of text if the message is of type 'TEXT' or if it is a media, the string representation of the own message id in and its extension: e.g. '/media/stringifiedId.png'





# **Vue.js Menu app component**



miki  
Hi

Type something...





Search...



miki

Hi

Hi  
14:11:54

# chat contact selection

Type something...





HandShake



miki



Search...



miki

Hi

Hi

14:11:54

# chat contact selection

## @click.native

Type something...





Search...



miki

Hi



miki

Hi  
14:11:54

# chat contact selection

@click.native

**propagation of the event to  
the main view AppContainer**

Type something...



# VUE setup & development



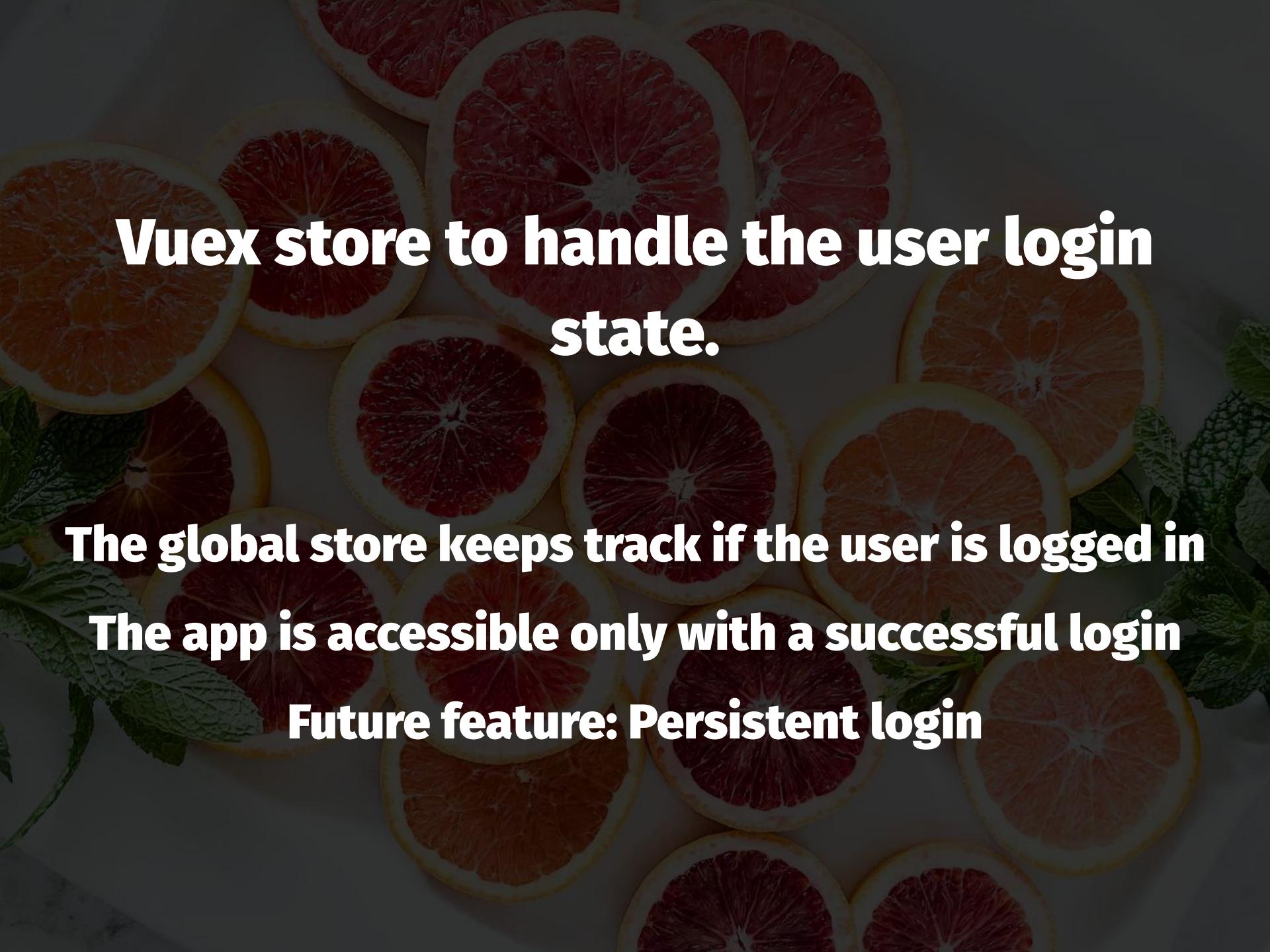
# **Setup the Vue app**

**Automatic build in public folder to serve it from Express.**

# **Quality check tools**

**Setup ESLint and Prettier to have a consistent code style.**

**Added Github action to check Vue code style.**

A background image featuring numerous slices of oranges and lemons, some with their green mint leaves still attached, creating a fresh and citrusy theme.

**Vuex store to handle the user login state.**

**The global store keeps track if the user is logged in**

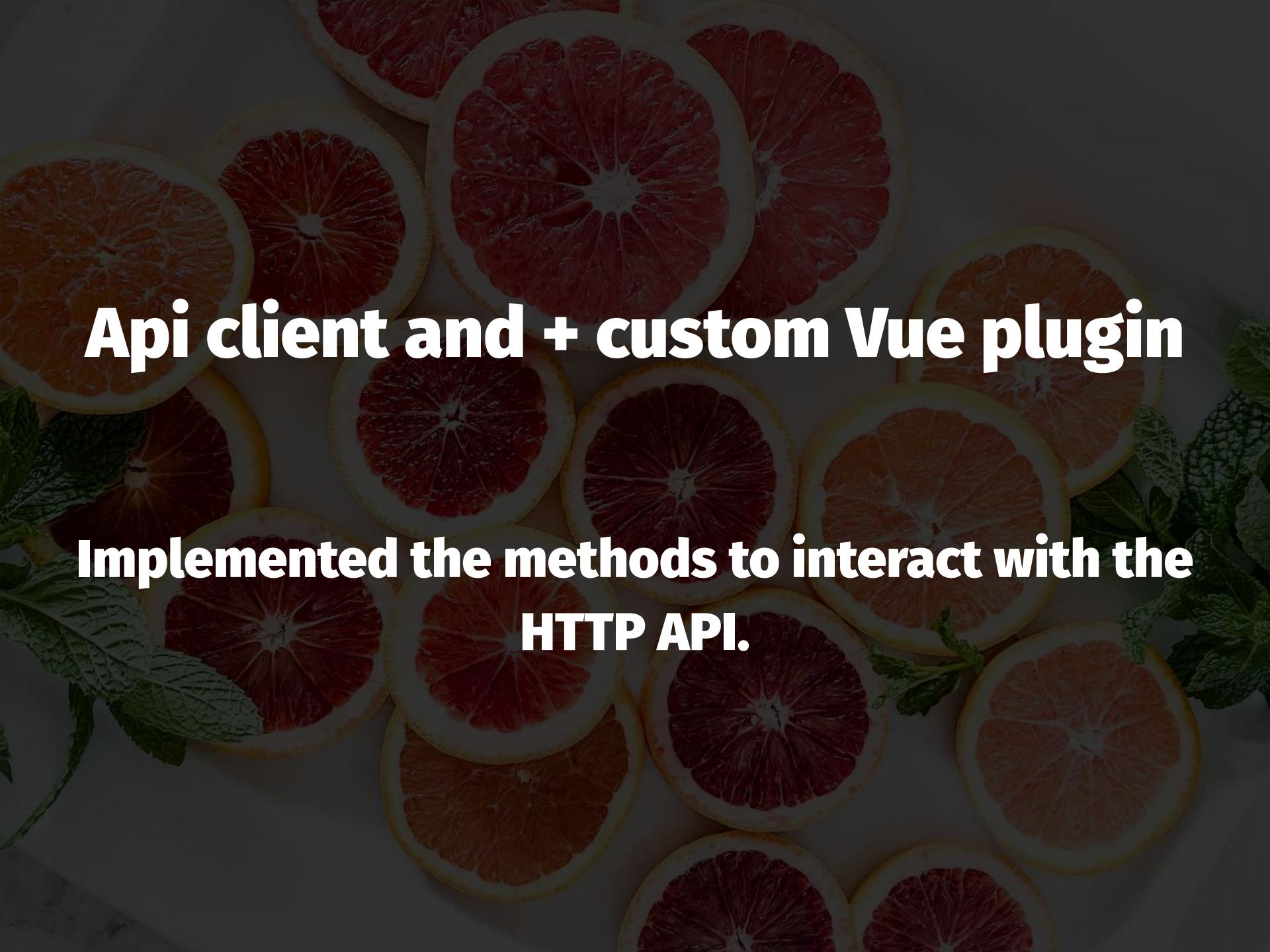
**The app is accessible only with a successful login**

**Future feature: Persistent login**

# **Setup Axios**

**Installed Axios in the Vue app**

**Created middleware to automatically refresh  
token.**

A background image featuring numerous slices of citrus fruits, likely blood oranges or grapefruits, arranged in a scattered pattern. The slices are vibrant red-orange with white pith and visible seeds. Interspersed among the fruit slices are several green mint leaves, adding a fresh, aromatic element to the composition.

# **Api client and + custom Vue plugin**

**Implemented the methods to interact with the  
HTTP API.**

# **Vue components development**

**Took care of advanced Vue topics to improve  
the components code.**

# **Dynamic user search feature**

**Created the app search bar to find new users to  
create a chat with.**

**Automatic users list update based on the  
server's response.**

# Pagination

**Created a limit in message length with a “Show more” link**

**Prevent long wait time to display big messages.**

# Bugs + Details

**Took care of crashy bugs both in client and server code.**

A close-up photograph of a person's hand pointing their index finger towards a detailed map of Southern California's highway network. The map shows a complex web of roads, including major interstate routes like I-10, I-5, and I-710, as well as state routes and local streets. The background is slightly blurred, emphasizing the hand and the map.

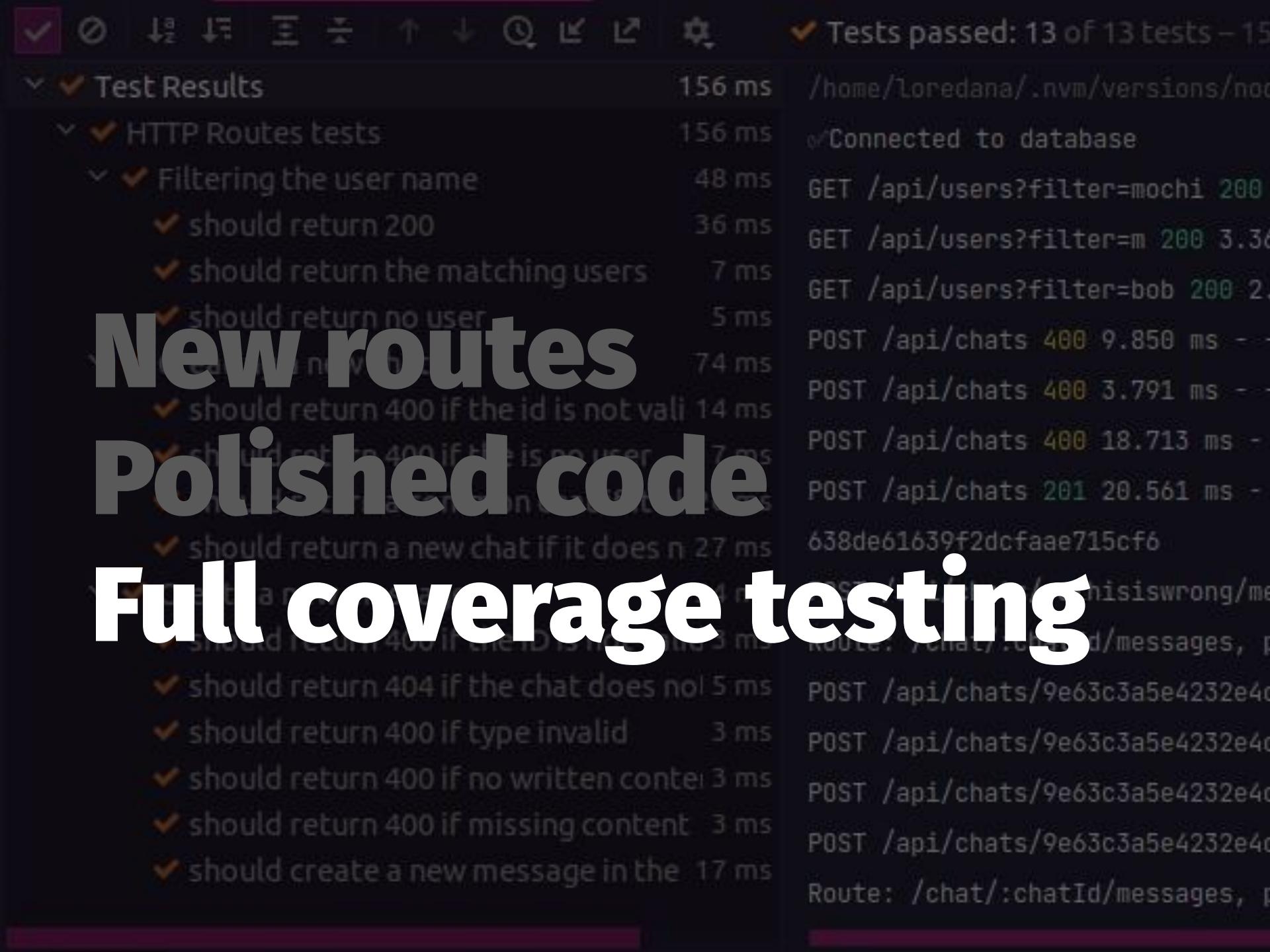
# routes

# New routes

✓	✗	↓↑	☰	↶	↷	⟳	⟳	⚙️	✓ Tests passed: 13 of 13 tests – 15
✗	✓	Test Results							/home/Loredana/.nvm/versions/node
✗	✓	HTTP Routes tests							✓ Connected to database
✗	✓	Filtering the user name							GET /api/users?filter=mochi 200
✓	✓	should return 200							GET /api/users?filter=m 200 3.36
✓	✓	should return the matching users							GET /api/users?filter=bob 200 2.
✓	✓	should return no user							POST /api/chats 400 9.850 ms -
✓	✓	should return 400 if the id is not valid							POST /api/chats 400 3.791 ms -
✓	✓	should return 400 if there is no user							POST /api/chats 400 18.713 ms -
✓	✓	should return a common chat if it already exists							POST /api/chats 201 20.561 ms -
✓	✓	should return a new chat if it does not exist							638de61639f2dcfaae715cf6
✗	✓	Create a new message							POST /api/chats/yothisiswrong/messages
✓	✓	should return 400 if the ID is not valid							Route: /chat/:chatId/messages, p
✓	✓	should return 404 if the chat does not exist							POST /api/chats/9e63c3a5e4232e40
✓	✓	should return 400 if type invalid							POST /api/chats/9e63c3a5e4232e40
✓	✓	should return 400 if no written content							POST /api/chats/9e63c3a5e4232e40
✓	✓	should return 400 if missing content							Route: /chat/:chatId/messages, p
✓	✓	should create a new message in the database							

# New routes Polished code

✓	✗	↓↑	☰	↶ ↷	⟳	⟳	⚙️	✓ Tests passed: 13 of 13 tests – 15
✗	✓	Test Results			156 ms	/home/Loredana/.nvm/versions/node		
✗	✓	HTTP Routes tests			156 ms	✓ Connected to database		
✗	✓	Filtering the user name			48 ms	GET /api/users?filter=mochi 200		
✓	should return 200				36 ms	GET /api/users?filter=m 200 3.36		
✓	should return the matching users				7 ms	GET /api/users?filter=bob 200 2.		
✓	should return no user				5 ms	POST /api/chats 400 9.850 ms -		
✓	should return 400 if the id is not valid				74 ms	POST /api/chats 400 3.791 ms -		
✓	should return 400 if the id is not valid				7 ms	POST /api/chats 400 18.713 ms -		
✓	should return 400 if the id is not valid				7 ms	POST /api/chats 201 20.561 ms -		
					638de61639f2dcfaae715cf6			
✓	should return a new chat if it does n				27 ms	POST /api/chats/yothisiswrong/me		
✗	✓ Create a new message				34 ms	Route: /chat/:chatId/messages, p		
✓	should return 400 if the ID is not valid				3 ms	POST /api/chats/9e63c3a5e4232e40		
✓	should return 404 if the chat does no				5 ms	POST /api/chats/9e63c3a5e4232e40		
✓	should return 400 if type invalid				3 ms	POST /api/chats/9e63c3a5e4232e40		
✓	should return 400 if no written conte				3 ms	POST /api/chats/9e63c3a5e4232e40		
✓	should return 400 if missing content				3 ms	POST /api/chats/9e63c3a5e4232e40		
✓	should create a new message in the				17 ms	Route: /chat/:chatId/messages, p		



New routes  
Polished code  
Full coverage testing

# backend-socket





On the first connection  
Login successfully

socket.join('All chats')



Retrieve contacts

Join in the chat

get\_message



retrieve messages

Write a message

post /chat/:chatID



message io.to(chatID)



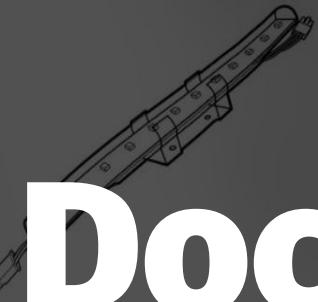
# Middleware Authentication

```
const cookie = require('cookie');
const authConstants = require('../constants/auth.constants');

// This middleware is used to check if the user is authenticated
// and if it is, it adds the user id to the socket object
const authMiddleware = async (socket, next) => {
    const jwtCookieName = authConstants.JWT_COOKIE_NAME;
    const cookies = cookie.parse(socket.request.headers.cookie);
    const jwtToken = cookies[jwtCookieName];
    try {
        const {userId} = await verifyJWT(jwtToken);
        socket.userId = userId;
        next();
    } catch (e) {
        console.log(e);
        next(new Error('Authentication error'));
    }
};

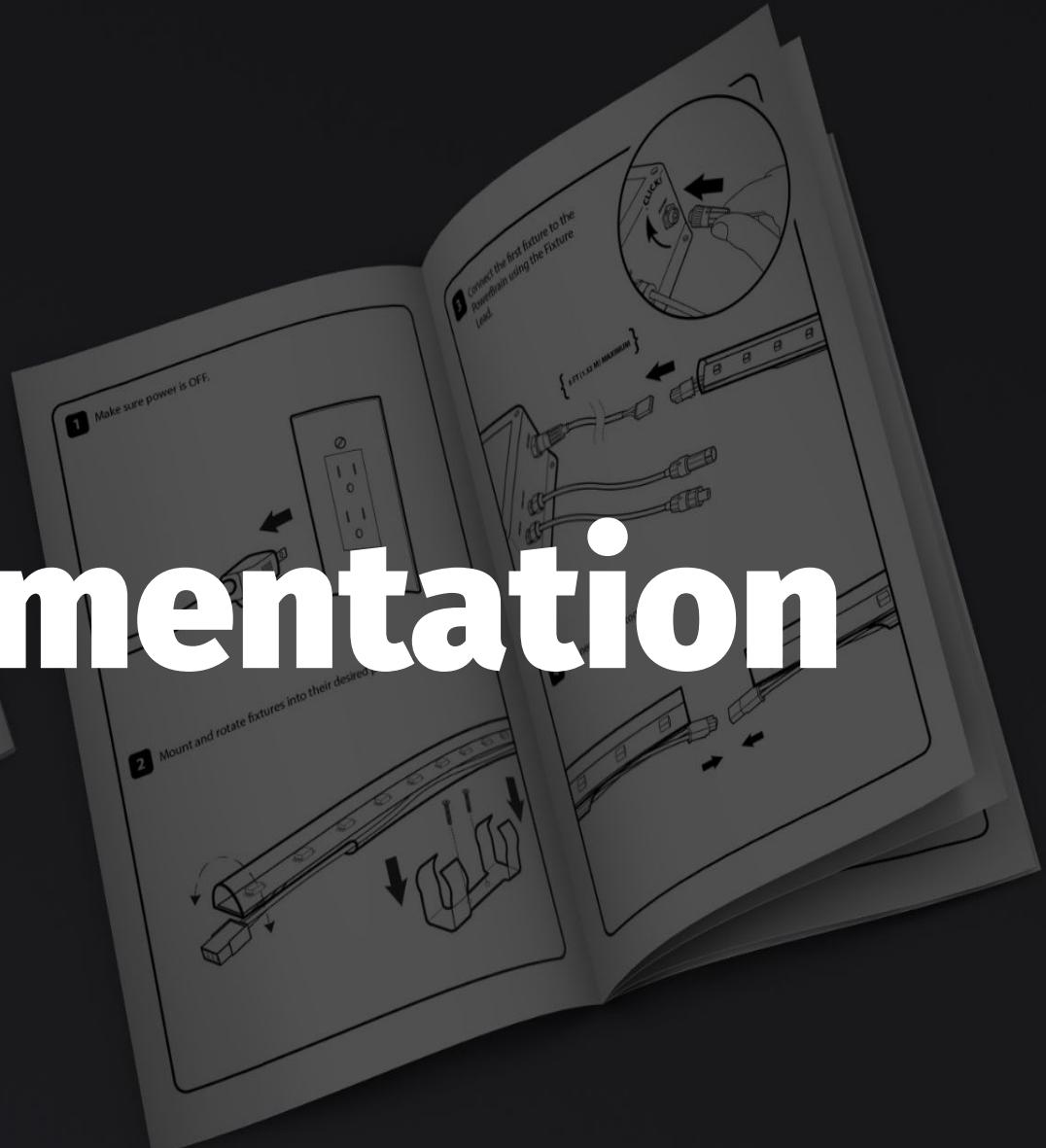
module.exports = {
    authMiddleware,
};
```

COLORLINC



illumvision

# Documentation



# Developer Guide

 slides	added milestone 1 slides in pdf
 ER.png	Added docs UML for dataclasses
 UML.png	Added docs UML for dataclasses
 er.mdj	Added docs UML for dataclasses
 github.md	Renamed docs, added structure.md
 layout.md	Renamed docs, added structure.md
 routes.md	Update routes.md
 socket.md	Update socket.md
 structure.md	Added ER link and fixed .gitignore

# Routes HTTP

---

## GET

---

### /users?filter=userstring

The client request an object with all the users in an array that match the name (or part of it) with the filter value `userstring`.

The `email`, `hashed password` and `chats` fields for the `User` are omitted from the User structure given back as response.

The field `_id` is leaved with the preceding underscore to internally emphasize that it needs to be converted to an `ObjectId`.

```
{
  "users": [
    {
      "_id": "123abc098",
      "name": "userstr",
    }
  ],
}
```

## POST

---

### /chat

Create a chat if it doesn't already exist within its members:

Server creates the chat document from the Mongoose model and saves it to db.

**side effect: socket** Server emits `chat:create` to the online connected new members (event client issuing the request) of the new chat with chat object from `Chat` class.

Server sends back an object containing the new chat id to the client who issued the request.



handshake Connect View Monitoring Browse Collections ... FREE SHARED

**Enhance Your Experience**  
For production throughput and richer metrics, upgrade to a dedicated cluster now!

**Upgrade**

**R 0**  
**W 0**  
Last 6 hours  
100.0/s

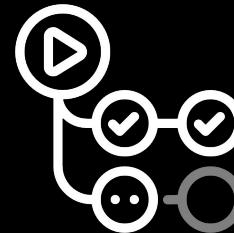
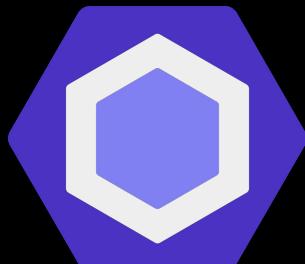
**Connections 2.0**  
Last 6 hours  
2.0

**In 25.4 B/s**  
**Out 183.2 B/s**  
Last 6 hours  
183.2 B/s

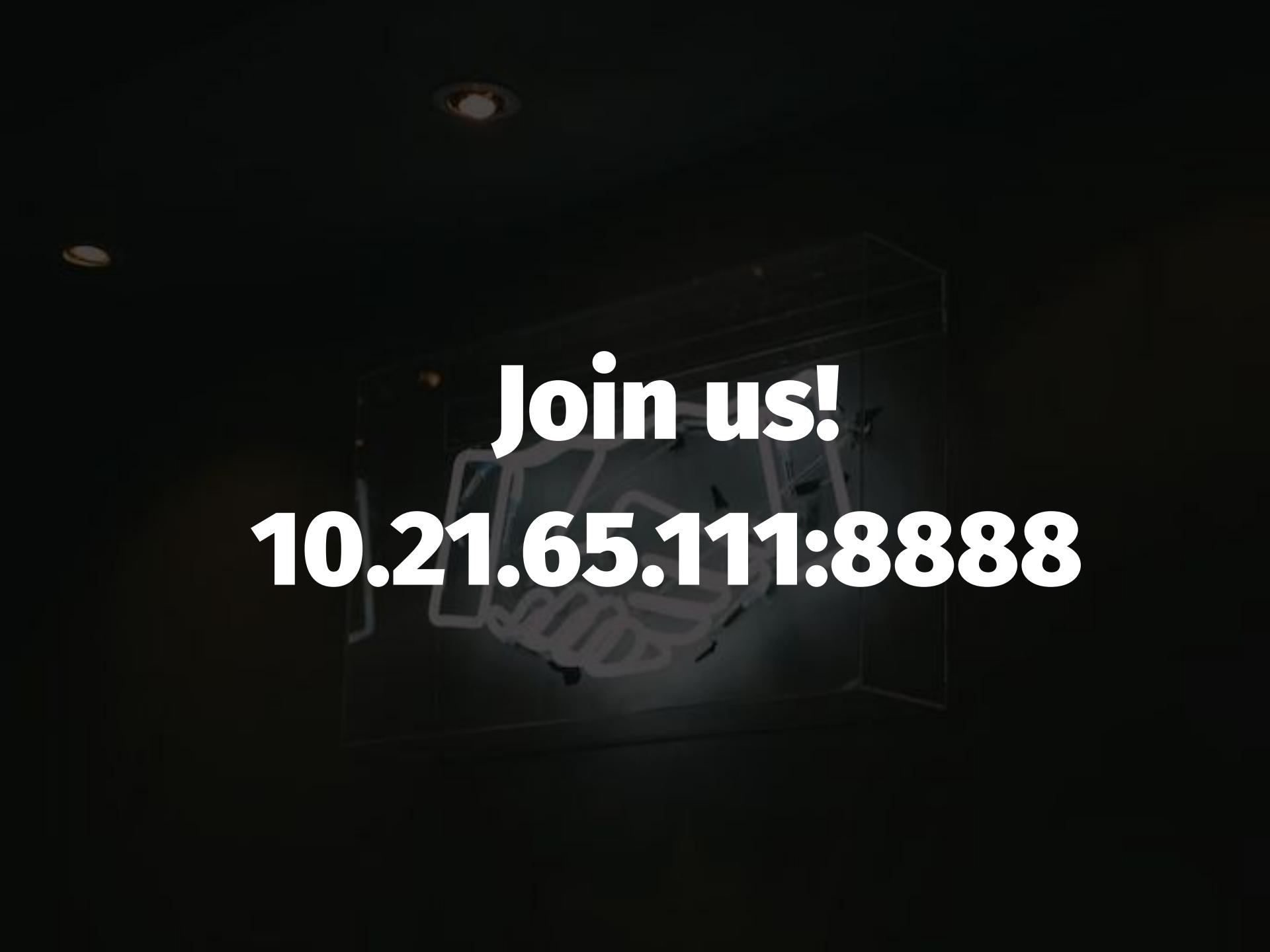
**Data Size 213.0 KB**  
Last 23 hours  
512.0 MB

VERSION	REGION	CLUSTER TIER	TYPE	BACKUPS	LINKED APP SERVICES	ATLAS SEARCH
5.0.14	AWS / Frankfurt (eu-central-1)	M0 Sandbox (General)	Replica Set - 3 nodes	Inactive	None Linked	Create Index

# Tools



A X 1 O S

The background of the image is a dark, low-light photograph of a building's exterior. A bright, glowing entrance or window area is visible in the center, casting a glow on the surrounding wall. There are some circular lights and architectural details visible.

**Join us!**

**10.21.65.111:8888**

# Thank you!

Dalle Rive Michele  
d'Atri Sofia  
De Filippo Loredana  
Jeferson  
Tafta Nicolò

See you at the next milestone!

